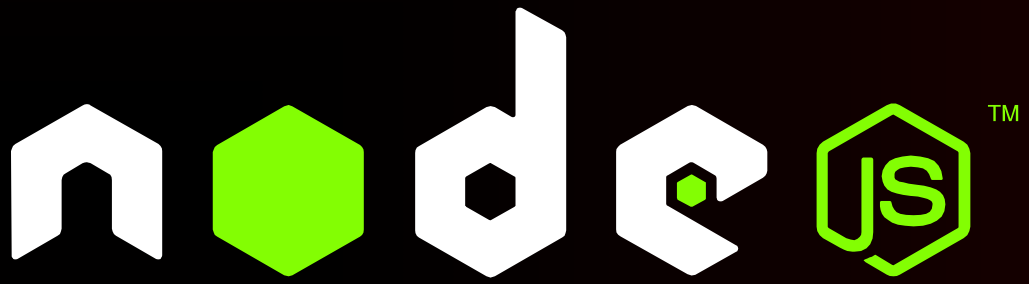
ON HARMONY

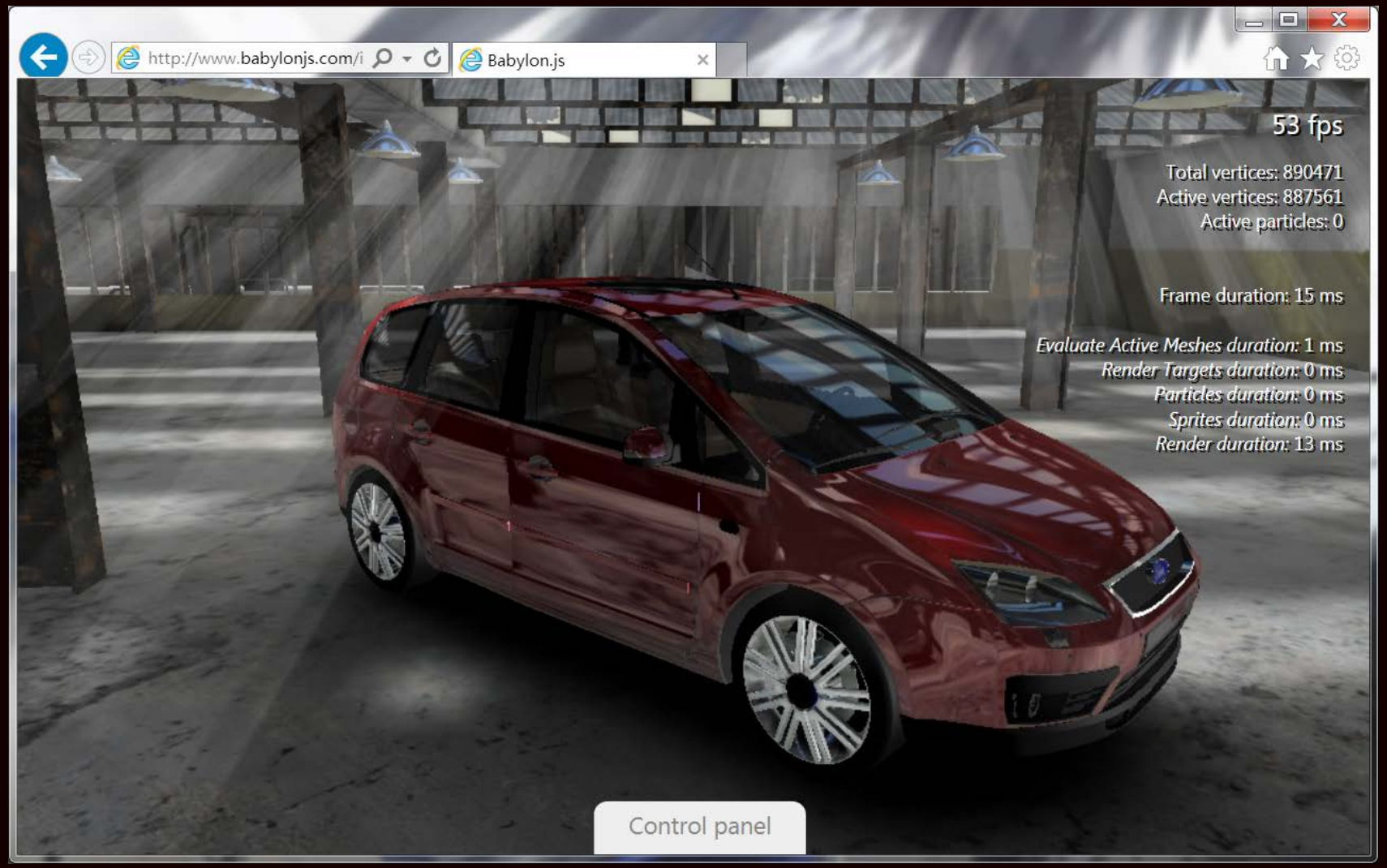
ON HARMONY

- Belleve Invis
- Node.js + 形式語言理論
- 計算化學 + 分子設計
- Github: be5Invis
- mail: belleve@typeof.net

JavaScript 建築的美麗世界


HTML


node
js
TM


http://www.babylonjs.com/i
Babylon.js
53 fps
Total vertices: 890471
Active vertices: 887561
Active particles: 0
Frame duration: 15 ms
Evaluate Active Meshes duration: 1 ms
Render Targets duration: 0 ms
Particles duration: 0 ms
Sprites duration: 0 ms
Render duration: 13 ms
Control panel

Atwood's law

「*Any application that can be written in JavaScript, will eventually be written in JavaScript*」

語言是一切的基礎

ECMA-262

缺陷

「設計模式」

```
function Class(Superclass, fn) {
    var proto = Object.create(Superclass.prototype)
    fn.call(proto, Superclass)
    if (proto.constructor instanceof Function) {
        var ctor = proto.constructor
        delete proto.ctor
    } else {
        var ctor = function () {}
    }
    var Type = function () {
        ctor.apply(this, arguments)
    }
    Type.prototype = proto
    return Type
}
```

```
var Dog = Class(Animal, function(supre) {
    this.bark = ......
    this.constructor = function() {
        supre.apply(this, arguments)
    }
})
var kula = new Dog()
kula.bark()
```

改進 JavaScript 本身

- ES-Harmony
- altjs

Harmony == ECMA-262 ver.6

Let's estimate that Harmony is approved as ECMAScript 6 in mid-2012 and Internet Explorer 11 is released in early 2013 with support for all of Harmony's syntax. Five years after that, in 2018, the Google Apps team can drop support for Internet Explorer 11 and finally use Harmony syntax freely.

Amazon developers might need to wait an additional 5 years before they can use Harmony syntax. That's 2023!

——Peter Michaux

Let's estimate that ~~Harmony is approved as ECMAScript 6 in mid-2012~~ and ~~Internet Explorer 11 is released in early 2013 with support for all of Harmony's syntax~~. Five years after that, in 2018, the Google Apps team can drop support for Internet Explorer 11 and finally use Harmony syntax freely.

Amazon developers might need to wait an additional 5 years before they can use Harmony syntax. That's 2023!

——Peter Michaux

altjs：構造可以編譯到 JavaScript 的新語言

Typescript

patrisika

……或者擴展 JavaScript

wind.js

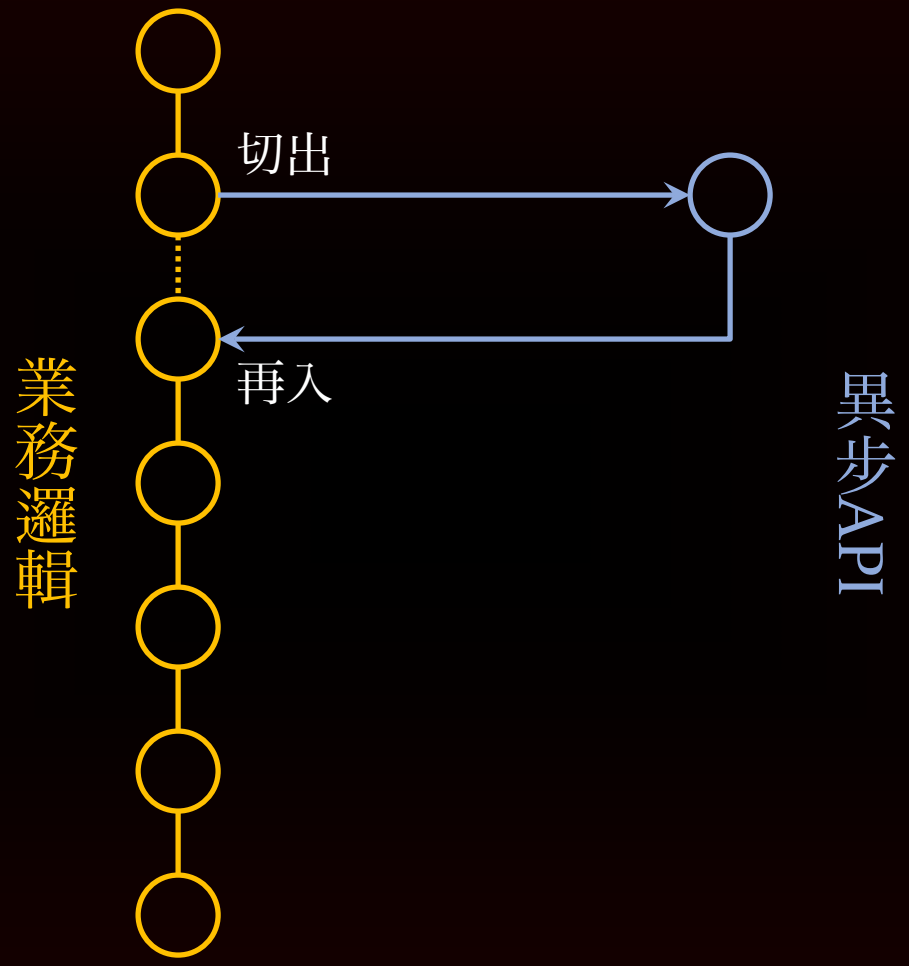

sweet.js

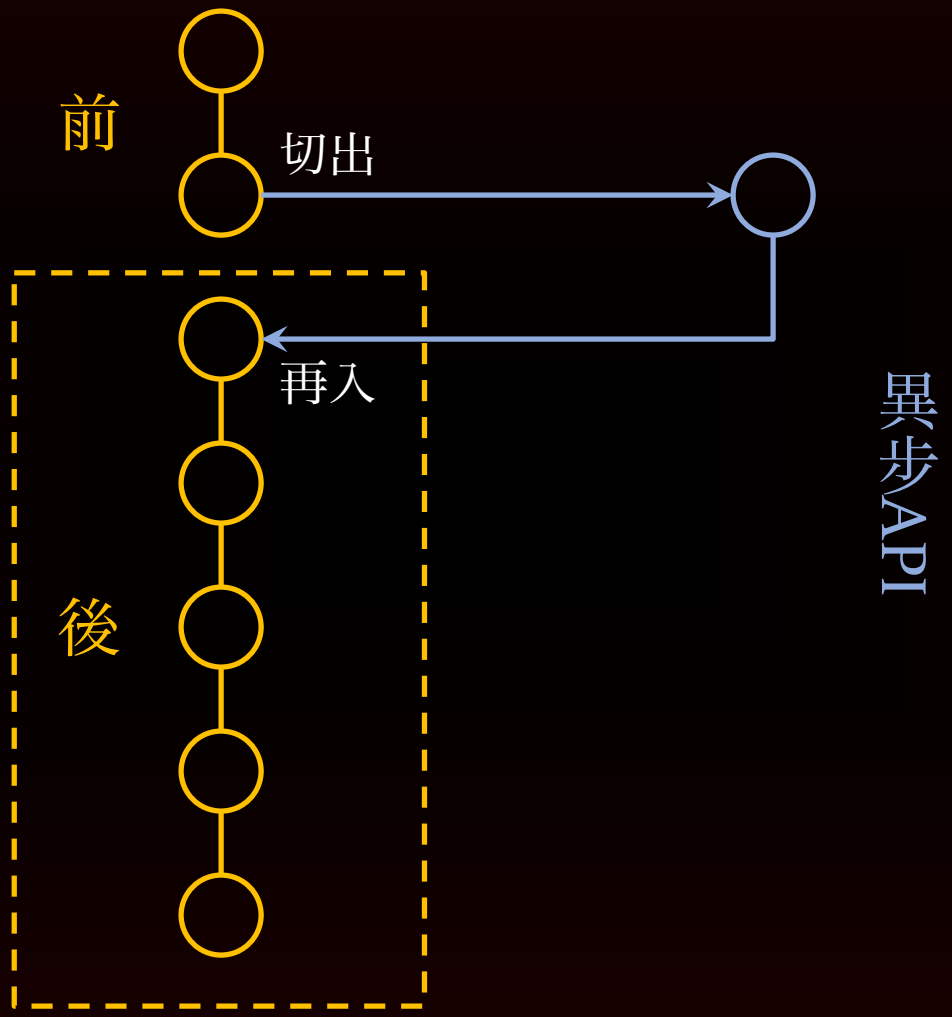
//

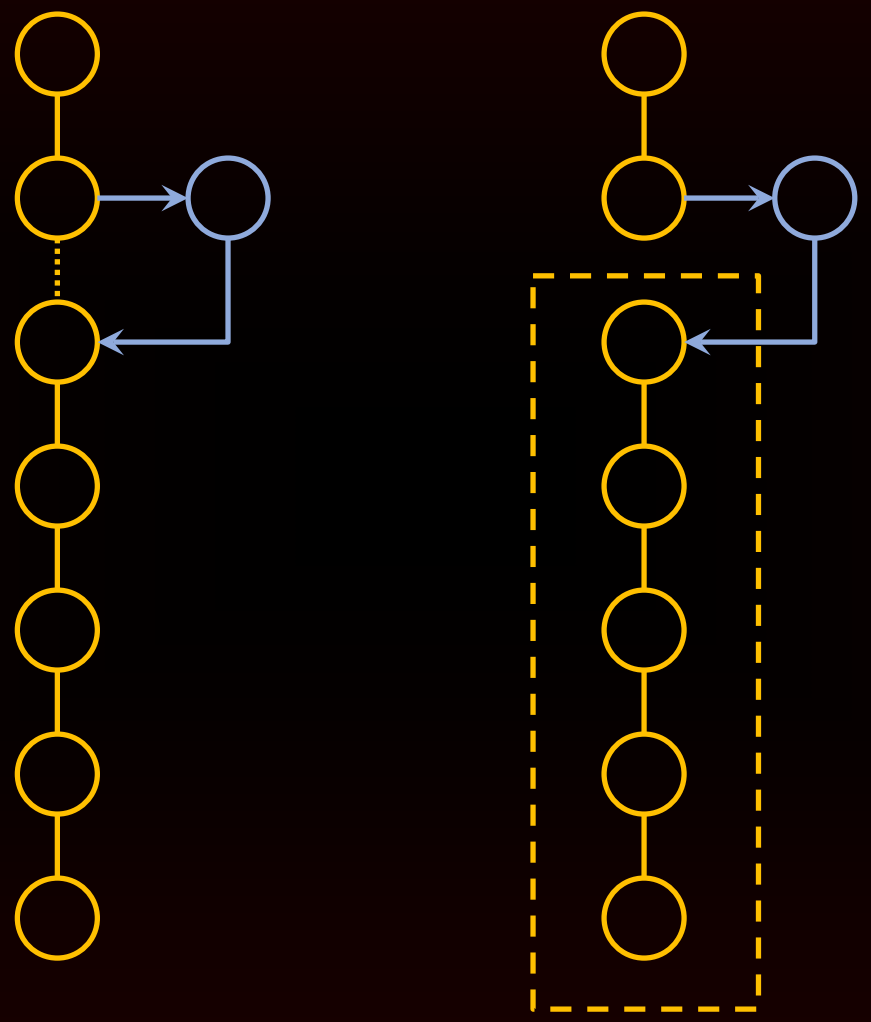
控制流抽象

```
op1(function(result1){
    op2(function(result2){
    })
})
```

```
var result1 = op1()
var result2 = op2()
```


業務邏輯
切出
再入
異步API


前
切出
再入
後
異步API



A continuation is an abstract representation of the control state of a computer program. A continuation reifies the program control state, i.e. the continuation is a data structure that represents the computational process at a given point in the process' execution; the created data structure can be accessed by the programming language, instead of being hidden in the runtime environment.

```
eval(Wind.compile('async', function() {
    var result1 = $await(op1)
    var result2 = $await(op2)
})
```

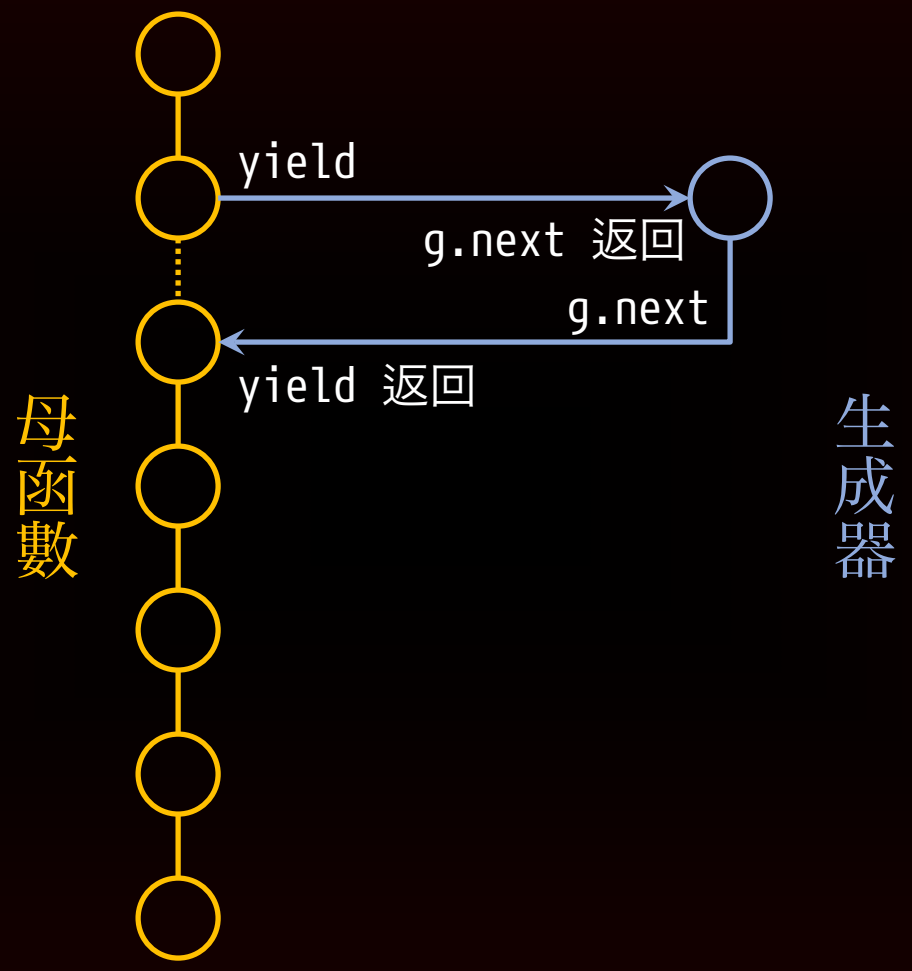
```
function*() {
    var result1 = yield op1
    var result2 = yield op2
}
```

```
function* bar(x) {
    x++;
    var y = yield x;
    yield y / 2;
}

var g = bar(1);
g.next();  // -> {value: 2, done: false}
g.next(8); // -> {value: 4, done: false}
g.next();  // -> {value: undefined, done: true}
```

```
function* bar(x) {
    x++;
    var y = yield x;
    yield y / 2;
}

var g = bar(1);
g.next();  // -> {value: 2, done: false}
g.next(8); // -> {value: 4, done: false}
g.next();  // -> {value: undefined, done: true}
```

```
function* bar(x) {
    x++;
    var y = yield x;
    yield y / 2;
}

var g = bar(1);
g.next();  // -> {value: 2, done: false}
g.next(8); // -> {value: 4, done: false}
g.next();  // -> {value: undefined, done: true}
```

8


yield
g.next 返回
g.next
yield 返回
母函數
生成器

```
for(var item of list) {
    console.log(item)
}
```

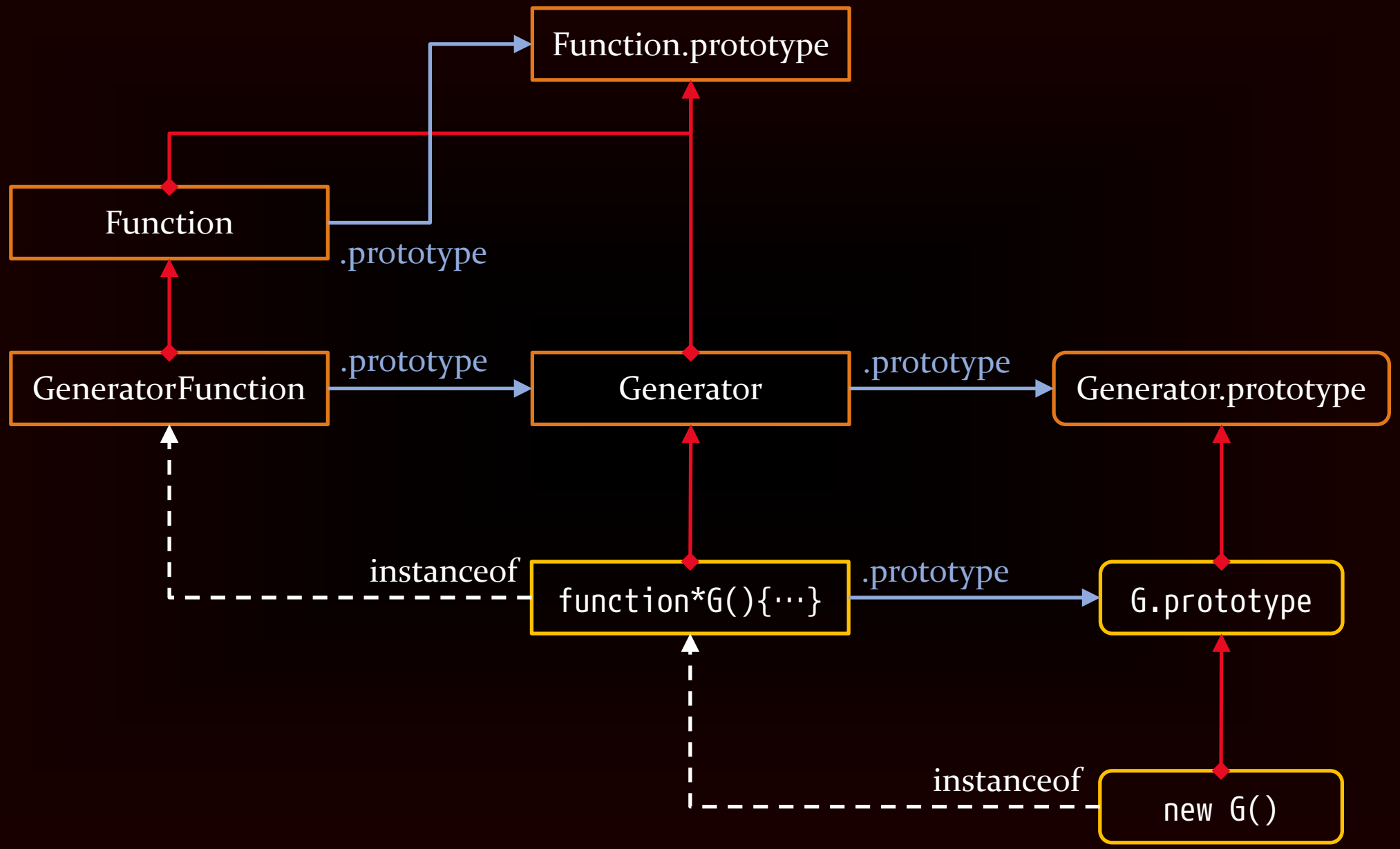
```
var iter = list[@@iterator]()
while(true) {
    var iv = iter.next();
    if(iv.done) break;
    var item = iv.value
    ......
}
```

```
Array.prototype[@@iterator] = function*(){
    for(var j = 0; j < this.length; j++) yield this[j]
}
Array.prototype.entries = function*(){
    for(var j = 0; j < this.length; j++)
        yield [j, this[j]]
}

Generator.prototype[@@iterator] = function(){
    return this
}
```

```
var nums = [1, 2, 3, 4, 5]
var evens = [for(x of nums) x * 2]
// [2, 4, 6, 8, 10]
var even_iter = (for(x of nums) x * 2)
for (var y of even_iter) {
    console.log(y) // 2, 4, 6, 8, 10
}
```

```
var async = function(G){
    return function(...args){ return function(callback) {
        var g = new G(...args)
        var step = function(ex, val){
            var done, task;
            if(ex) {
                {done, value: task} = g.throw(val);
            } else {
                {done, value: task} = g.next(val);
            }
            if(!done) { task(step) }
            else if(callback) { callback(task) }
        };
        step();
    } }
}
```

```
async(function*() {
    yield op1
    yield op2
})
```


Function.prototype
Function
.prototype
GeneratorFunction
.prototype
Generator
.prototype
Generator.prototype
instanceof
function*G(){…}
.prototype
G.prototype
instanceof
new G()

- 控制流抽象增強
- 簡化異步程序設計
- 簡化列表處理

結構化數據

- 二進制數據分析
- 簡化信息提取
- 增強反射機能

```
[a, b] = f(x)

var t = f(x);
a = t[0]
b = t[1]
```

```
var file = {name: name, size: size, mtime: mtime}
//
var name = file.name
var size = file.size
var mtime = file.mtime
```

```
var file = {name: name, size: size, mtime: mtime}
//
var {name: name, size: size, mtime: mtime} = file
```

```
var file = {name: name, size: size, mtime: mtime}
//
var {name, size, mtime} = file
```

```
var {name, size = 0, mtime} = file
```

```
var { op: a, lhs: { op: b }, rhs: c } = getASTNode()
for (let [[k], {salary: s}] of database) {···}
```

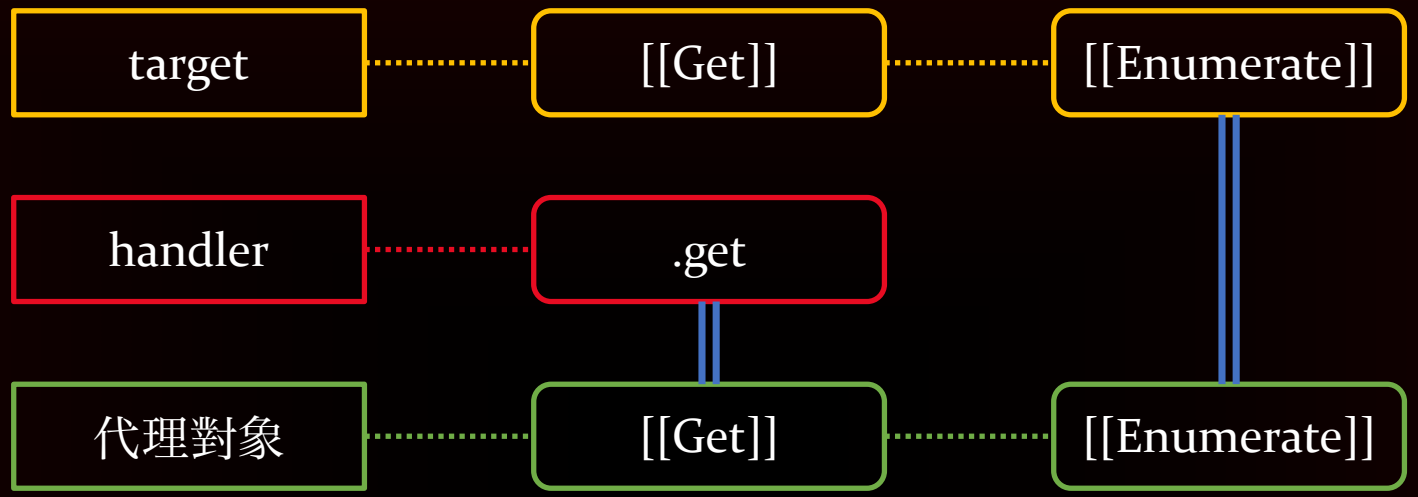
```
var run = function (func, ...args) {
    return func(...args)
    // equalivent to func.apply(null, args)
}

run(function(x, y){return x + y}, 1, 2) // -> 3

var fold = function ([head, ...rear], f, init) {
    if(!rear.length) return f(head, init)
    else return f(head, fold(rear, f, init))
}
```

反射 API + 代理對象

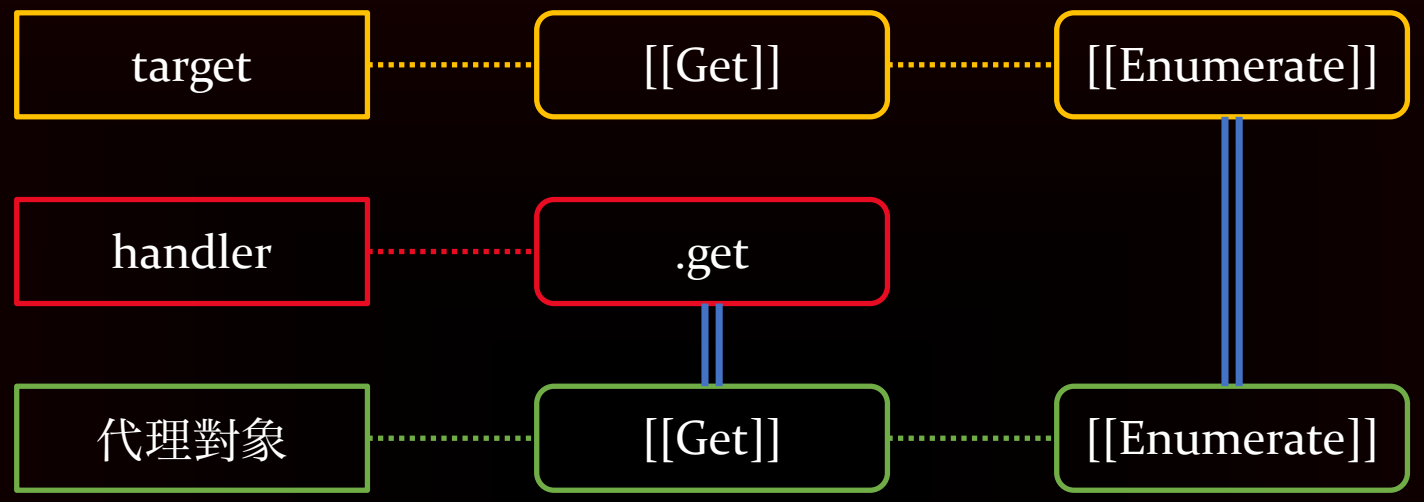
- [[GetPrototypeOf]] ( )
- [[SetPrototypeOf]] (V)
- [[IsExtensible]] ( )
- [[PreventExtensions]] ( )
- [[GetOwnProperty]] (P)
- [[DefineOwnProperty]] (P, Desc)
- [[HasProperty]] (P)
- [[Get]] (P, Receiver)
- [[Set]] (P, V, Receiver)
- [[Invoke]] (P, ArgumentsList, Receiver)
- [[Delete]] (P)
- [[Enumerate]] ()
- [[OwnPropertyKeys]] ( )
- [[Call]] (thisArgument, argumentsList)
- [[Construct]] Internal Method

- Reflect.getPrototypeOf
- Reflect.setPrototypeOf
- Reflect.isExtensible
- Reflect.preventExtensions
- Reflect.getOwnPropertyDescriptor
- Reflect.defineProperty
- Reflect.has
- Reflect.get
- Reflect.set
- Reflect.invoke
- Reflect.deleteProperty
- Reflect.enumerate
- Reflect.ownKeys

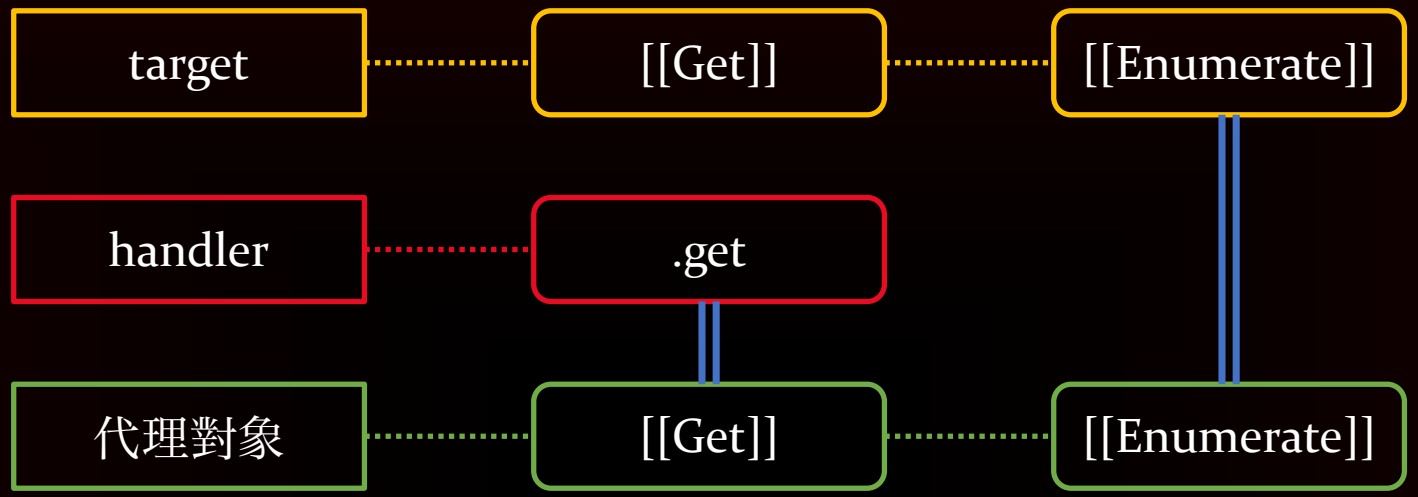
```
proxy = new Proxy(target, handler)
```




```
proxy.[[Enumerate]] = function() {
    return target.[[Enumerate]]()
}
```




```
proxy.[[Get]] = function(name, receiver) {
    return handler.get(proxy, name, receiver)
}
```




```
(proxy[name]) ⇔ proxy.[[Get]](name, proxy)
              ⇔ handler.get(proxy, name, proxy)
```

```
var RangeCheckedArray = function (lo, hi) {
    return new Proxy([], {
        get: function (target, name, recv) {
            if (+name >= lo && +name < hi) {
                return Reflect.get(target, name, recv)
            } else {
                throw "Out of range"
            }
        },
        ......
    })
}
```

```
set: function (target, name, value, receiver) {
    if (+name >= lo && +name < hi) {
        return Reflect.set(target, name, value, receiver)
    } else {
        throw "Out of range"
    }
},
// handler invoke(target, name, args, receiver):
// invoke method target[name] with args as arguments
// and receiver as "this"
invoke: function (target, name, args, receiver) {
    return this.get(target, name).apply(receiver, args)
},
......
```

語法糖 class

- 類定義由一系列方法定義組成
- `constructor(){}` 「方法」作爲構造器
- 每個類可繼承一個基類
- 使用 super 關鍵字訪問基類的方法

```
class Dog extends Animal {
    constructor(name) {
        super(name);
    }
    isFoodEatable(food) {
        return super(food) && food instanceof Meat
    }
}
```

```
super.x        Reflect.get(superProto, 'x', this)
super.x = y    Reflect.set(superProto, 'x', y, this)
super.f(x)     Reflect.invoke(superProto, 'f', [x], this)
new super(x)   new superClass(x) // 於構造器內
super(x)       // 由所在方法決定語義，爲以下兩者之一
               superClass.call(this, x) // 構造器內
               Reflect.invoke(superProto, 'm', [x], this) // 方法 m 內
```

```
class Dog extends Animal {
    constructor(name) {
        Animal.call(this, name);
    }
    isFoodEatable(food) {
        return Animal.prototype.isFoodEatable.call(this, food)
            && food instanceof Meat
    }
}
```

- ArrayBuffer 和 DataView
- 解構賦值：快速提取信息
- 反射 API 和代理對象：增強反射和客製化能力
- class：簡化 OOP

```
for(let x of list) { f(x) }
// is equalivent to
for(var t of list) { (function(x){ f(x) }(t)) }
```

```
tag`The total is ${total} (${total * 1.05} with tax)`
// is equalivent to
tag(["The total is ", " (", " with tax)"],
    total, total * 1.05)
```

```
list.map(x => x * 2)
```

Set + Map

Weakset + Weakmap

```
module 'math' {
    export function sum(x, y) { return x + y }
    export var pi = 3.141593;
}
import {sum, pi} from 'math';
alert("2π = " + sum(pi, pi));
```

加入 es-discuss

es-discuss@Mozilla.org

Q+A

Live Long and Prosper.